Lingüística computacional

Representación y procesamiento de la gramática léxico-funcional

Xavier Gómez Guinovart
Seminario de Lingüística Informática, Universidad de Vigo

<sli@uvigo.es>

Resumen: *en este artículo se presentan los fundamentos teóricos de la gramática léxico-funcional y los problemas de su representación formal y de su procesamiento computacional. Trataremos de mostrar cómo la implementación de teorías lingüísticas permite comprobar exhaustivamente las consecuencias de los modelos teóricos, facilitando la detección de errores en los análisis, fortaleciendo la consistencia global de los modelos y estimulando el rigor formal en las descripciones de fenómenos lingüísticos.*

Palabras clave: *procesamiento del lenguaje natural, sintaxis computacional*

1. Introducción

La gramática léxico-funcional (**LFG** o *lexical-functional grammar*) es un modelo lingüístico computacional desarrollado a finales de la década de 1970 en la Universidad de Stanford en California, como fruto de la colaboración entre la lingüista Joan Bresnan (de tradición generativista) y el informático Ronald Kaplan, conocido por sus trabajos pioneros sobre análisis morfosintáctico automático mediante redes de transición aumentadas. El objetivo de la LFG consiste en la elaboración de un modelo altamente formalizado del lenguaje humano, un modelo del lenguaje computacionalmente preciso, psicológicamente realista y de orientación lingüística lexicista y funcional [2] [3].

En el terreno de la sintaxis, la LFG se caracteriza por mantener que la estructura sintáctica debe representarse al menos en dos niveles: uno correspondiente a la estructura de constituyentes (representada habitualmente mediante un diagrama arbóreo) y otro correspondiente a la estructura funcional. La estructura de constituyentes (denominada, en este modelo, «estructura-c») se describe formalmente mediante una gramática sintagmática independiente del contexto y constituye la base del procesamiento fonológico de la oración. Por su parte, la estructura funcional (o «estructura-f») representa en LFG las funciones gramaticales y otras propiedades gramaticales de base funcional presentes en la oración, como la concordancia, el tiempo, la definitud, el control o la anáfora. La estructura-f adopta la forma de una matriz no ordenada de rasgos, donde cada rasgo está compuesto por un parámetro o propiedad lingüística y por el valor que adopta dicha propiedad en la estructura analizada. A modo de ejemplo, véase (en `EF_1`) una estructura-f posible para la oración «El alumno aprobó los exámenes», dejando la explicación de algunos de los detalles de su representación para los siguientes apartados.

`EF_1:`

```
[
PRED = 'aprobar <(" SUJ)  (^ OBJ)>'
TPO = +PAS
SUJ = [
       PRED = 'alumno'
       DEF = +
       NÚM = -PL
       GÉN = -FEM
       PERS = 3
       ]
OBJ= [
       PRED = 'examen'
       DEF = +
       NÚM = +PL
       GÉN = -FEM
       PERS = 3
       ]
]
```

Los rasgos de las estructuras-f pueden tener valores simples o atómicos (como `PRED` y `TPO`, en `EF_1`) y valores complejos o recursivos (como `SUJ` y `OBJ`), donde el valor es, a su vez, otra matriz de rasgos incrustada en la más amplia. Como en otros formalismos contemporáneos que utilizan estructuras complejas de rasgos para representar la información lingüística, las estructuras-f de la LFG se construyen y manipulan mediante el procedimiento matemático de la «unificación» [1] [4] [6]. La unificación de dos estructuras de rasgos (**ER**) equivale a la unión de todos los rasgos contenidos en ambas, pero sólo tiene lugar si las dos estructuras no contienen valores incompatibles. Si la información contenida en una de las ER resulta contradictoria con la contenida en la otra, la unificación no puede llevarse a cabo. Véanse unos ejemplos sencillos de unificación de ER: la `ER_1` y la `ER_2` unifican en `ER_4`, pero la `ER_3` no puede unificar ni con `ER_1` ni con `ER_2` por contener un valor contradictorio con `ER_1` para la propiedad de `NÚM` y un valor contradictorio con `ER_2` para la propiedad de `TPO`.

`ER_1:`

```
[  SUJ = [  NÚM = +PL ]  ]
```

`ER_2:`

```
[
SUJ = [  PERS = 3 ]
```

```
TPO = −PAS
]
```

ER_3:
```
[
SUJ = [ NÚM = −PL ]
TPO = +PAS
]
```

ER_4:
```
[
SUJ = [
       PERS = 3
       NÚM = +PL
       ]
TPO = −PAS
]
```

En LFG, la descripción de las estructuras-c y de las estructuras-f se realiza simultáneamente, mediante reglas sintagmáticas ampliadas con descripciones funcionales que definen de manera implícita las características de las estructuras-f. Veámoslo con un ejemplo:

R_1:
```
O  ♦
       SN
       (″ SUJ) = ⌡

       SV
       ″ = ⌡
```

La regla R_1 describe, en su parte sintagmática (es decir, en el nivel superior de la regla), las relaciones de orden y constitución entre los sintagmas nominal (SN) y verbal (SV) componentes de la estructura de la oración (O), mientras que las anotaciones funcionales al pie de la regla indican cómo participa en la estructura-f la información funcional contenida en los nudos específicos del árbol de constituyentes. Los vectores hacia arriba y hacia abajo se refieren a la estructura-f que corresponde al nudo de la estructura-c construido por la regla. El vector ″ alude a la estructura-f del nudo madre y el vector ⌡ a la estructura-f del propio nodo. Así pues, las ecuaciones funcionales adscritas a los nudos de la parte derecha de la regla indican que la estructura-f correspondiente al nudo SN se unifica con la estructura-f del rasgo SUJ correspondiente al nodo O, y que las estructuras-f correspondientes a los nodos SV y O se unifican en una estructura común.

Ilustraré a continuación el funcionamiento de este mecanismo de descripción sintáctica mediante el análisis de la oración «Juan corre» a partir de las reglas aumentadas de R_2, R_3 y R_4 y de las entradas léxicas de EL_1 y EL_2:

R_2:
```
O  ♦
       SN
       (″ SUJ) = ⌡

       SV
       ″ = ⌡
```

R_3:
```
SN  ♦
       N
       ″ = ⌡
```

R_4:
```
SV  ♦
       V
       ″ = ⌡
```

EL_1:
```
Juan N
(″ PRED) = ‘Juan’
(″ DEF) = +
(″ NÚM) = −PL
(″ GÉN) = −FEM
(″ PERS) = 3
```

EL_2:
```
corre V
(″ PRED) = ‘correr <(″ SUJ)>’
(″ NÚM SUJ) = −PL
(″ PERS SUJ) = 3
(″ TPO) = −PAS
```

En EC_1 recojo la estructura-c derivada de la aplicación de las reglas, indicando las anotaciones funcionales en los nodos; mientras que en ER_5 a ER_9 especifico las estructuras-f correspondientes a los cinco nodos implicados.

EC_1:
```
      O_9
    ___|____
  SN_6   SV_8
    |      |
   N_5    V_7
  Juan   corre
```

ER_5 = ER_6:
```
[
PRED = ‘Juan’
DEF = +
NÚM = −PL
GÉN = −FEM
PERS = 3
]
```

ER_7 = ER_8:
```
[
PRED = ‘correr <(″ SUJ)>’
TPO = −PAS
SUJ = [
       NÚM = −PL
       PERS = 3
       ]
]
```

ER_9:
```
[
PRED = ‘correr <(″ SUJ)>’
TPO = −PAS
SUJ = [
```

```
        PRED = 'Juan'
        DEF = +
        NÚM = −PL
        GÉN = −FEM
        PERS = 3
        ]
]
```

A continuación, veremos cómo se pueden representar y procesar algunos análisis concretos de la LFG en un formalismo que nos permita algunas implementaciones informáticas sencillas, con vistas a la ilustración práctica de las aplicaciones de la teoría.

2. Procesamiento de gramáticas léxico-funcionales

Como acabamos de ver, la LFG es una teoría lingüística que emplea diversos mecanismos formales para expresar sus análisis. En este sentido, y a nivel general, es preciso distinguir entre teoría lingüística y formalismo lingüístico, reservando el segundo término para los lenguajes artificiales utilizados por los modelos teóricos para la representación formal de sus análisis. El formalismo utilizado por la teoría LFG ha sido adaptado en diversas ocasiones para facilitar su procesamiento computacional, generalmente, empleando el lenguaje de programación Prolog como intermediario entre el formalismo adaptado y el propio ordenador. Las primeras adaptaciones computacionales del formalismo LFG tuvieron lugar a mediados de la década de 1980, iniciando una línea de investigación a la que han contribuido de manera destacada los profesores Juan Carlos Ruiz Antón (de la Universidad Jaume I) y Joseba Abaitua (de la Universidad de Deusto).

Existen actualmente numerosos formalismos lingüísticos aptos para el procesamiento computacional de modelos basados en la unificación de rasgos, pero no limitados de manera definitiva a ninguna teoría lingüística específica, entre los que cabe mencionar **PATR** (*Parsing and Translation*), **ALE** (*Attribute Logic Engine*), **TFS** (*Typed Features Structures*) o **DATR**. Estos formalismos implementables, que permiten su adaptación a distintos modelos teóricos, suelen recibir la denominación de «lenguajes de programación lingüística», distinguiéndose terminológicamente de los «formalismos lingüísticos teóricamente motivados», como los utilizados por la LFG y por otras teorías basadas en la unificación. Veremos a continuación, un ejemplo de implementación de LFG en PATR y, más concretamente, en PC-PATR, un sistema de programación lingüística para PC elaborado y distribuido libremente en Internet <ftp://ftp.sil.org/software/dos/> por el Summer Institute of Linguistics de Dallas que permite implementar gramáticas escritas en PATR [5].

PATR es un formalismo desarrollado originalmente por Stuart Shieber en la Universidad de Stanford a mediados de la década de 1980 para escribir gramáticas sintagmáticas formadas por reglas independientes del contexto y aumentadas con estructuras de rasgos sobre las que opera la unificación. Una gramática descrita en PATR consta de un conjunto de reglas y de un léxico. El léxico contiene las palabras de la lengua junto con sus propiedades relevantes y proporciona los elementos que sirven para remplazar los símbolos terminales en las reglas de estructura sintagmática. Las reglas sintagmáticas permiten establecer cuáles son las estructuras de constituyentes de una lengua. Su formato general corresponde al de las reglas independientes del contexto, es decir, A ♦ α, siendo A un símbolo no terminal de la gramática, y α una cadena de cualquier tipo formada por símbolos terminales o no terminales de la gramática. La única peculiaridad de PATR a este respecto consiste en que, al escribir una gramática PATR, se debe indicar el inicio de cada regla mediante la clave Rule («regla», en inglés), como hacemos en G_1.

G_1:

```
Rule O -> (SN) SV
Rule SN -> {(Det) N (SP)} / Pron
Rule SP -> P SN
Rule SV -> V (SN)
```

En esta gramática hay un símbolo inicial o categoría máxima del análisis sintáctico O; los símbolos no terminales o categorías sintácticas O, SN, SV y SP; y los símbolos terminales o categorías léxicas Det, N, P y V (no analizables ya mediante ninguna otra regla). Algunos recursos notacionales de PATR que se puede observar en el ejemplo son:

1. Los constituyentes opcionales se representan entre paréntesis.
2. La alternancia de constituyentes se representa separando mediante una barra inclinada los constituyentes o grupos de constituyentes alternativos.
3. Las llaves se emplean para agrupar grupos de constituyentes para que la indicación de la alternancia no sea ambigua.

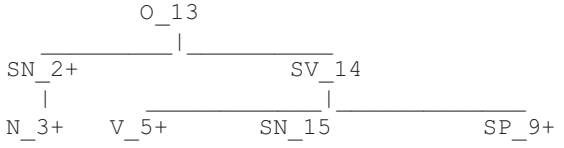
Siguiendo con el ejemplo, utilizaremos G_1 conjuntamente con un léxico extremadamente simple, donde las entradas no contienen más información que la forma léxica y su categoría sintáctica. En PC-PATR, un léxico así se definiría mediante los códigos «\w» y «\c», como se ilustra en L_1.

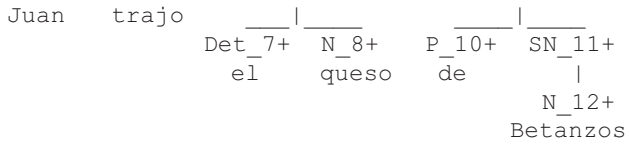
L_1:

```
\w el \c Det
\w Juan \c N
\w trajo \c V
\w Betanzos \c N
\w queso \c N
\w de \c P
```

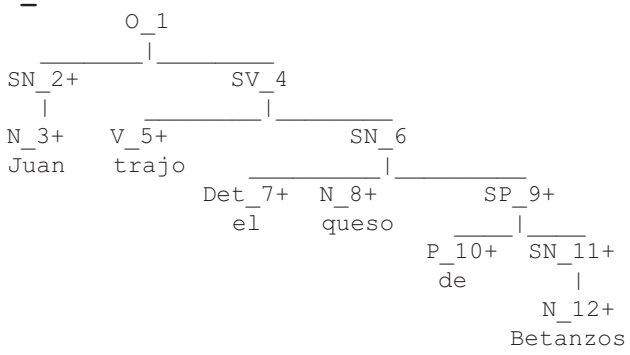
Con G_1 y L_1, podemos utilizar PC-PATR para analizar la estructura de constituyentes de una oración sintáctica y semánticamente ambigua como "Juan trajo el queso de Betanzos", con los resultados que se recogen en EC_2 y EC_3.

EC_2:





EC_3:



En estos árboles, elaborados automáticamente por PC-PATR como resultado del análisis, los números identifican los nodos y constituyen índices que apuntan a sus propiedades, como veremos inmediatamente; mientras que las cruces indican los constituyentes que poseen las mismas propiedades en los dos análisis y, por tanto, pueden referirse mediante un único índice.

Una vez vista la manera de describir estructuras de constituyentes empleando el formalismo PATR, veamos cómo se pueden expresar en este formalismo las estructuras funcionales postuladas por la LFG. Decíamos anteriormente que las reglas de PATR pueden aumentarse con condiciones expresadas mediante la unificación de estructuras de rasgos. La unificación de rasgos puede utilizarse bien para comprobar la identidad exigida en dos propiedades gramaticales asociadas a distintos constituyentes, como en el caso de la concordancia; bien para copiar propiedades gramaticales de un constituyente a otro, por ejemplo, del núcleo de una construcción a su nodo dominante inmediato. PATR permite asociar estructuras de rasgos con los elementos léxicos y con los constituyentes de las reglas. Los rasgos deben especificarse uniendo con un signo de igualdad cada propiedad lingüística con su valor, y escribiendo la propiedad entre corchetes triangulares.

En ER_10 se muestra cómo se pueden representar en PATR la estructura de rasgos expresada antes en forma de matriz de ER_3 .

ER_10:

```
<SUJ NÚM> = -PL
<TPO> = +PAS
```

Este ejemplo nos sirve además para ilustrar la forma de expresar en PATR los rasgos complejos, como el de SUJ de ER_10. Como se ve, cada valor final de la estructura de rasgos se expresa mediante el camino o secuencia de propiedades que llevan a él. Con estos fundamentos, veamos en G_2 cómo se expresaría en el formalismo PATR la gramática léxico-funcional descrita en R_2, R_3 y R_4, con las ecuaciones funcionales de la LFG expresadas mediante la unificación de estructuras de rasgos.

G_2:

```
Rule O -> SN SV
       <O EF> = <SV EF>
       <O EF SUJ> = <SN EF>
Rule SN -> N
       <SN EF> = <N EF>

Rule SV -> V
       <SV EF> = <V EF>
```

Obsérvese que, en PATR, el primer símbolo de la secuencia o camino de rasgos de las ecuaciones corresponde a la categoría del nodo a la que se asocia la estructura de rasgos. Obsérvese también que utilizamos el rasgo denominado EF para describir la estructura funcional asignada a cada constituyente de la estructura-c; es decir, el valor de EF en cada nodo será la estructura de rasgos correspondiente a la descripción funcional del nodo en cuestión. Así, la regla PATR correspondiente al SV indica que la estructura-f correspondiente al SV y la estructura-f correspondiente al V unifican; mientras que la regla correspondiente a O indica, en primer lugar, que las estructuras-f de O y SV unifican, y en segundo lugar, que también unifican la estructura-f correspondiente al SN y la estructura de rasgos correspondiente a la propiedad SUJ que se encuentra dentro de la estructura-f de O. Veamos ahora en L_2 la representación en PATR del léxico LFG descrito antes en EL_1 y EL_2 para esta gramática.

L2:

```
\w Juan \c N
\f <EF PRED> = 'Juan'
<EF DEF> = +
<EF NÚM> = -PL
<EF GÉN> = -FEM
<EF PERS> = 3
\w corre \c V
\f <EF PRED> = 'correr'
<EF FUNCS> = SUJ
<EF SUJ NÚM> = -PL
<EF SUJ PERS> = 3
<EF TPO> = -PAS
```

En cuanto a esta descripción léxica, obsérvese en L_2 que las propiedades lingüísticas distintas de la forma léxica y la categoría, expresadas como ecuaciones funcionales en LFG, se especifican en PATR tras el código «\ f»; que los caminos de PATR siguen el orden inverso (de la propiedad general a la particular) a los del formalismo de la LFG (que van de la más particular a la más general); y que utilizamos el rasgo denominado FUNCS para establecer en PATR las funciones subcategorizadas descritas junto al significado léxico como valor de PRED en LFG.

Por último, veamos el resultado del análisis automático realizado por PC-PATR de la frase «Juan corre» a partir de la gramática y el léxico que acabamos de presentar en G_2 y L_2. Este resultado incluye la estructura-c (EC_4) y la estructura-f de la oración (bajo el nodo etiquetado O_1), más las estructuras-f asignadas a los constituyentes durante la deducción en paralelo de la estructura-c y la estructura-f de la oración. Obsérvese que los nodos del diagrama arbóreo

están relacionados con sus correspondientes descripciones funcionales mediante un índice numérico, como ya avanzamos anteriormente. Obsérvese también que PC-PATR sitúa automáticamente, en la estructura de rasgos correspondiente a cada constituyente, un rasgo CAT con el valor de su categoría sintáctica, y en los nodos terminales, un rasgo adicional LEX con el valor de su forma léxica.

```
EC_4:
      O_1
    ___|____
  SN_2   SV_4
    |      |
   N_3    V_5
  Juan   corre


O_1:
[  EF:      [  PRED:   'correr'
               TPO:    -PAS
               FUNCS:  SUJ
               SUJ:      [  PRED:   'Juan'
                            DEF:    +
                            GÉN:    -FEM
                            NÚM:    -PL
                            PERS:   3 ] ]
   cat:    O ]

SN_2:
[  EF:      [  PRED:   'Juan'
               DEF:    +
               GÉN:    -FEM
               NÚM:    -PL
               PERS:   3 ]
   cat:    SN ]

N_3:
[  EF:      [  PRED:   'Juan'
               DEF:    +
               GÉN:    -FEM
               NÚM:    -PL
               PERS:   3 ]
   cat:    N
   lex:    Juan ]

SV_4:
[  EF:      [  PRED:   'correr'
               TPO:    -PAS
               FUNCS:  SUJ
               SUJ:      [  NÚM:    -PL
                            PERS:   3 ] ]
   cat:    SV ]

V_5:
[  EF:      [  PRED:   'correr'
               TPO:    -PAS
               FUNCS:  SUJ
               SUJ:      [  NÚM:    -PL
                            PERS:   3 ] ]
   cat:    V
   lex:    corre ]
```

A pesar de ser un ejemplo extremadamente sencillo, podemos observar en él las dos aplicaciones mencionadas de la unificación, es decir la comprobación de rasgos (en este caso, para la comprobación de la concordancia de número y persona entre el sujeto y el verbo) y la copia de rasgos (empleada en esta ocasión para la elevación de propiedades nucleares y para la construcción de la estructura funcional de manera paralela a la estructura de constituyentes).

3. Conclusiones

La posibilidad de expresar modelos computacionales del lenguaje mediante formalismos lingüísticos adecuados, y la aplicación de estos modelos a cualquier nivel de descripción lingüística, supone diversas ventajas para la lingüística teórica. Por una parte, permite comprobar exhaustivamente las consecuencias de las hipótesis teóricas implementadas, facilitando así la detección de errores e incoherencias en la descripción de fenómenos lingüísticos. Por otra parte, permite comprobar la consistencia global de los modelos teóricos elaborados, al proporcionar un medio efectivo y práctico para observar la interacción de todos los componentes del modelo postulado.

4. Referencias

[1] Abeillé, Anne (1993). *Les novelles syntaxes: grammaires d'unification et analyse du français*. París: Colin.
[2] Bresnan, Joan (2001). *Lexical-functional syntax*. Oxford: Blackwell.
[3] Kaplan, Ronald y Joan Bresnan (1982). «Lexical-functional grammar: a formal system for grammatical representation». En Bresnan, Joan (ed.) (1982). *The mental representation of grammatical relations*, pp. 173-281. Cambridge: The MIT Press.
[4] Kay, Martin (1992). «Unification». En Rosner, Michael y Roderick Johnson (eds.). *Computational linguistics and formal semantics*, pp. 1-29. Cambridge: Cambridge University Press.
[5] McConnel, Stephen (1998). *PC-PATR reference manual: a unification based syntactic parser*. Dallas: Summer Institute of Linguistics.
[6] Shieber, Stuart (1986). *An introduction to unification-based approaches to grammar*. Stanford: Center for the Study of Language and Information.